A HYBRID APPROACH FOR REVERSE ENGINEERING GUI MODEL FROM MOBILE APPS FOR AUTOMATED TESTING

SALIHU IBRAHIM ANKA

A thesis submitted in
fulfillment of the requirement for the award of the
Doctor of Philosophy in Information Technology

Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia

AUGUST, 2017

PTTA UTHM
PERPUSTAKAAN TUNKU TUN AMINAH

To my beloved mother and my late father.

PTTA UTHM
PERPUSTAKAAN TUNKU TUN AMINAH

ACKNOWLEDGEMENT

My heartiest gratitude to Allah (SWT) for making this journey a successful one. Firstly, I would like to express my sincere appreciation to my supervisor, Prof. Dr. Rosziati Ibrahim, for her support, guidance and criticism during my PhD study. She inspired me greatly to work on this research. I appreciate her constructive ideas and suggestions in the course of this research.

My special appreciation goes to the members of staff at the Faculty of Computer Science and Information Technology (FSKTM), and the Centre for Graduate Studies, Universiti Tun Hussein Onn Malaysia (UTHM) for providing a conducive learning environment for students. I must also acknowledge the office of Research, Innovation, Commercialization and Consultancy Management (ORICC) for supporting me during the course of this research. I would also like to thank my research colleagues at the faculty and friends for their insightful discussions, collaborations and support.

Last but not the least, a very special thanks you to my family: my mother, my brothers Sani Salihu Anka, Rufa'i Ahmad Sani Yarima and Dr. Mukhtar Salihu Anka for the never ending support and encouragement, and to my wife for their unconditional support and understanding through my good and bad times.

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

ABSTRACT

The past few years have experienced a massive transformation in personal computing where mobile devices are rapidly replacing traditional computers for an increasing number of users. This has impacted the area of software development through the growing sector of mobile applications for mobile devices. Like the traditional software applications, mobile apps must also be tested to ensure they behave correctly. Graphical User Interface (GUI) testing has been an effective means of validating GUI software particularly Android mobile applications (mobile apps). However, it still suffers a strong challenge about how to explore event sequence in the GUIs. Researchers and practitioners have proposed several approaches and tools for automated testing of mobile apps. Most of the approaches reverse engineer a model of an application under test and use it for the creation of test cases. However, the models generated by existing approaches are not comprehensive due to inability to explore application's behaviour extensively. This study proposes a technique based on hybrid approach for the systematic exploration of mobile apps' events which exploit the capabilities of both static and dynamic approaches while trying to improve application's state exploration and the generation of a high-quality model from a mobile app. A static analysis was performed on application's bytecode to extract events supported by an app and use the extracted events to dynamically explore an app at run-time. The hybrid approach was implemented in our tool called AMOGA (Automated Model Generator for Android). AMOGA is developed in Java programming language and was validated using real world mobile apps. Results of the experimental evaluation show that AMOGA has 45%-93% coverage across the 13 apps. In comparison to other existing tools, the result shows that AMOGA achieves better coverage than all the tools with a difference of above 4% on all the apps.

ABSTRAK

Sejak kebelakangan ini, transformasi secara besar-besaran telah berlaku dalam pemilikan komputer peribadi di mana peranti mudah alih dengan cepat telah menggantikan komputer tradisional dari segi peningkatan jumlah pengguna. Oleh itu, peningkatan sektor aplikasi mudah alih untuk peranti mudah alih telah memberi kesan kepada bidang pembangunan perisian. Seperti aplikasi perisian tradisional, aplikasi mudah alih juga perlu diuji untuk memastikan ia berfungsi dengan baik. Pemeriksaan terhadap Antara muka pengguna grafik (GUI) merupakan cara yang berkesan dalam mengesahkan perisian GUI terutamanya aplikasi mudah alih Android. Walau bagaimanapun, cabaran yang kuat masih dialami tentang bagaimana untuk meneroka urutan acara dalam GUI. Penyelidik dan pengamal telah mencadangkan beberapa pendekatan dan alat-alat untuk pemeriksaan aplikasi mudah alih secara automatik. Kebanyakan pendekatan kejuruteraan balikan telah digunakan terhadap sesuatu model yang masih di bawah ujian untuk mewujudkan kes-kes pemeriksaan. Walau bagaimanapun, model yang dihasilkan oleh pendekatan yang sedia ada tidak menyeluruh kerana disebabkan ketidakupayaan untuk meneroka tingkah laku aplikasi secara meluas. Kajian ini telah mencadangkan pendekatan hibrid untuk penerokaan yang lebih sistematik untuk aplikasi mudah alih dengan mengeksploitasi keupayaan kedua-dua pendekatan statik dan dinamik ketika cuba untuk meningkatkan penerokaan keadaan aplikasi dan penjanaan model berkualiti tinggi dari aplikasi mudah alih. Satu analisis statik telah dilakukan ke atas “bytecode” aplikasi untuk mengeluarkan acara yang disokong oleh aplikasi dan menggunakan peristiwa-peristiwa yang telah diekstrak untuk meneroka aplikasi secara dinamik semasa run-time. Pendekatan hibrid telah dilaksanakan pada alat kami yang dikenali sebagai AMOGA (Penjana Model Automatik untuk Android). AMOGA dibangunkan dalam bahasa pengaturcaraan Java dan disahkan menggunakan aplikasi mudah alih dunia nyata. Keputusan penilaian eksperimen

menunjukkan bahawa AMOGA mempunyai 45% - 93% liputan di seluruh 13 subjek. Berbanding dengan alat-alat lain yang sedia ada, hasilnya menunjukkan bahawa AMOGA mencapai liputan yang lebih baik daripada semua alat-alat dengan perbezaan melebihi 4% pada semua subjek.

PTTA UTHM
PERPUSTAKAAN TUNKU TUN AMINAH

TABLE OF CONTENTS

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

LIST OF TABLES

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

LIST OF FIGURES

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

PTTA UTHM
PERPUSTAKAAN TUNKU TUN AMINAH

LIST OF ABBREVIATIONS

AMOGA	-	Automated Model Generator for Android
App	-	Application
CFG	-	Control Flow Graph
DFG	-	Data flow Graph
GATOR	-	Program Analysis Toolkit for Android
GUI	-	Graphical User Interface
EFG	-	Event flow Graph
MA	-	Mobile Application
MBT	-	Model-based Testing
RE	-	Reverse Engineering
SATG	-	Static Activity Transition Graph
UI	-	User Interface
VA	-	Validity Analysis
VT	-	Validity Threat
WTG	-	Windows Transition Graph

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

LIST OF PUBLICATIONS

Journal:

- Salihu, I.A. and R. Ibrahim, **A Hybrid Technique for Reverse Engineering GUI Model from Mobile Apps for Testing.** Journal of Telecommunication, Electronic and Computer Engineering (JTEC). (Accepted 2017)

- Salihu, I.A. and R. Ibrahim, **Reverse Engineering Mobile Apps for Model Generation Using a Hybrid Approach**. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 2016. 8(4): p. 1-5.

Proceeding:

- Salihu, I.A. and R. Ibrahim, **Systematic Exploration of Android apps' events for Automated Testing**, in 14th International Conference on Advances in Mobile Computing & Multimedia (MoMM2016). 2016, ACM. p. 295-305.

- Salihu, I.A. and R. Ibrahim, **Comparative Analysis of GUI Reverse Engineering Techniques**, in Advanced Computer and Communication Engineering Technology. 2016, Springer. p. 295-305.

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

CHAPTER 1

INTRODUCTION

The introduction of mobile devices such as smartphones and tablets, has brought a paradigm shift in personal computing where mobile devices are rapidly replacing traditional computers for an increasing number of users in several areas, such as access to emails, e-commerce, social networks etc. [1, 2]. Smartphones emerged in different platforms (i.e. operating systems) such as Android, iOS, Windows, blackberry etc. A recent study indicated that a total of 349 million smartphones were sold to users worldwide in the first quarter of 2016 and Android OS is leading with 78.8% market share [3].

The widespread of smartphones and tablets has been observed worldwide by the recent dominance of its mobile apps over the traditional desktop applications on the internet, particularly in the US and UK [4, 5]. Due to the popularity of these devices, the past few years have experienced an immense transformation in the area of software development which appeared through the growing sector of mobile applications for mobile devices (smartphones and tablets) to meet up with the respective needs of their users [6].

In view of the divergence in mobile platforms (e.g., iOS, Android, Windows 7, etc.), mobile apps are developed for different platforms using various programming languages such as java commonly used in Android, Object-C for iOS, Visual C++ for Windows etc. Currently there are around 1,868,470 mobile applications on Apple's store, with 1,686,257 on Google Play store, 206,284 on Windows Marketplace and 130,000 on Blackberry Apps World.

The development of mobile applications has significant impact from both economic and social perspectives. It has generated a revenue of $4.5 billion USD in 2009 and it was predicted that global applications business will be worth $77 billion

by 2017 [7-9]. The widespread of mobile devices have attracted the attentions of software developers and researchers about the quality of the applications. This is due to the fact that inefficiencies in performance of mobile apps can affect user experience and the application could be uninstalled by unsatisfied users.

The aim of this research is to identify the strength and weaknesses of current mobile apps testing approaches and techniques in order to propose an improved technique for testing mobile apps. Based on the literature, both academia and the industry have resorted to model-based testing approaches for testing mobile apps. Therefore, this study focuses on reverse engineering techniques and tools for automated model generation from mobile apps.

This chapter presents the background information of the research, the research motivation, the research objectives and scope as well as significance of the research.

1.1 Background of the Study

Mobile apps are software systems designed for mobile devices such as smartphones, tablets and other handheld devices. Though, mobile apps are initially simpler and smaller with less complex design architecture and having a small set of functionalities, they are now becoming more and more complex [7, 8]. Mobile apps developers are recently increasing the capacity, functionality and quality of mobile apps, and the request for more complex, rich and usable functionalities will ever keep increasing in the next future [10]. In view of this, the modern mobile apps are more usable and capable of supporting more user tasks [7] and they are also moving to more business-critical uses [11]. Therefore, the traditional software engineering approaches and techniques cannot be directly applied in context of mobile devices [12]. This is due to the complex architecture of mobile platforms. For example mobile devices user interfaces (UI) provide a new paradigm for new human-computer interaction sequences with multi-touch interfaces, QR code scanning, image recognition, augmented reality, etc., that have not been available in traditional software. Therefore it has not been previously explored in research and of which no established UI guidelines exist [12, 13].

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

Recently, mobile apps are used not only for entertainment or social networking but also in safety and critical domains, such as payment systems, mobile government, m-health initiatives, etc. [11, 14, 15]. The widespread reliance on mobile apps in everyday life poses significant concern on apps quality such as correctness, performance, and security [16-19]. Furthermore, the increase in complexity of mobile apps has brought several challenges for the software engineering researchers such as understanding apps behavior and testing [4, 20, 21].

According to Wasserman [11], one of the important software engineering challenges with mobile application development today is that of finding effective solutions for achieving non-functional qualities in mobile applications and defining suitable techniques and tools to support their testing. Testing automation specifically can play a significant role in assessing and improving the quality of mobile applications [10, 22, 23]. Therefore, there is a demand for software engineering techniques and tools to support the testing task for mobile apps [12, 14, 24].

With the recent development in mobile apps, manual testing can no longer suffice because it is often tedious, error-prone and insufficient to achieve high coverage [4, 25]. Test automation is becoming increasingly popular among the software engineering community in recent times [23]. Model-based testing (MBT) is an approach/technique for test automation where the test execution is automatically derived from the model of a system under test (SUT) [26]. It provides a notable improvement to conventional scripted testing by enhancing the creation of test scripts and test coverage of an application [27]. However, constructing the model manually is an error prone task and time consuming [28]. In order to benefit from model-based testing, there is a demand for techniques/tools to aid automated model generation from mobile apps. Automated model generation from software applications is specifically a reverse engineering process.

Reverse engineering is an act of analysing a software system to extract design and implementation information and abstracting the information in the form of a model for easy understanding by users [29]. It is gaining more popularity from the research community as an approach for reconstructing original specifications or design from software. It is used for various purposes other than software understanding such as software testing, checking software reliability, software reusability, security analysis, updating user interfaces, migration and porting user interfaces to new platforms [30, 31].

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

1.2 Research Motivation

Android applications are a subset of the more general class of event-driven systems. They are specifically event-driven Graphical User Interface (GUI) applications. The GUIs serves as the main source of interaction with the applications [32] and it is where most errors occur [16, 33-35]. Mobile apps, like all other software, must be adequately tested to make sure they function correctly [10]. However, the current testing techniques and tools for traditional application cannot be applied to mobile apps [12].

Recently, the demand for tools that automate the testing process of mobile apps has grown with a focus on model-based testing tools. Model-based testing receives as input a model of an application to output a set of test cases that can be run to test an application. However, building these models fully automatically for the Android apps remains challenging [36-38]. Furthermore, the models generated by existing techniques are incomplete due to the inability to explore mobile apps' state extensively [24]. This is necessary because the application's behaviour can be affected by the states transitions during interaction with the application.

Some of the challenges are due to the nature of the Android platform (framework-based) where many of the behaviour of an application resides in the Android framework such as context events, and the support for multiple user gestures such as pinching by the android platform. Furthermore, mobile apps support a wide variety of events such as user events and system (context) events [23]. Recognizing these challenges, researchers and practitioners have proposed several techniques and tools to reverse engineer models from a mobile app through static or dynamic analysis or both (hybrid). Nonetheless, the models suffer from inadequate quality to cover the GUI behaviour comprehensively.

To improve the quality of GUI model generated from mobile apps for model-based testing, several reverse engineering techniques for automated model generation from mobile apps have emerged. Most of these techniques are typically based on dynamic approaches such as Android GUITAR [39], AndroidRipper [36, 40] and MCrawlT [41]. However, the information extracted by pure dynamic approach is incomplete [42]. As such the models generated by these techniques are incomplete due to the limitations of dynamic analysis [27, 43, 44]. Furthermore, dynamic exploration of an application in existing techniques is usually based on a crawler

guided with DFS (Depth First Search) or BFS (Breadth First Search) algorithms. The standard DFS algorithm usually needs to backtrack several times to make sure all paths (edges) in a graph are covered. With a dense graph, redundancy becomes more prevalent causing an increase in computational time [4, 41]. BFS is also not good for weighted di-graphs because it only considers the number of edges but does not account for the weight of the edges.

Tools that combined both static and dynamic approaches were proposed recently to improve coverage and the quality of model generated from mobile apps such as Orbit tool [4] and A^3E [20]. Nonetheless, the models generated by these tools are incomplete. For example, several mobile apps behaviours are not captured in the models such as complex gesture and lifecycle events. Therefore, test cases that are derived from such models will only provide low code coverage of an application. One of the limitations of these approaches is that their static analysis is not comprehensive, because it does not model the changes to the window stack and does not considers analysis of callbacks and determines ordering constraints between them [45, 46]. Hence, they have not provided an optimal solution for the Android apps in terms of events coverage. Therefore, there is the need for further contributions in improving the effectiveness of the techniques in order to generate high quality models for testing Android apps.

The problems are therefore summarized as follows:

1. The models generated by existing techniques are incomplete due in ability to explore app's state extensively [24].

2. Most of the existing techniques [10, 25, 36, 47] are based on dynamic analysis to explore and extract app's behaviour. However, the GUI information extracted by pure dynamic approach is incomplete [42].

3. The most challenging issue with any dynamic analysis technique includes the way and order in which GUI events are found and fired [27].

This study contributes to the emerging area of research on mobile apps by proposing a new technique based on a hybrid approach for the generation of a high quality model from Android app. As highlighted by previous researchers, one of the

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

most challenging issues with any dynamic analysis technique includes the way and order in which GUI events are found and fired [27, 43, 48]. The proposed approach extracts mobile app's events statically by analysing its bytecode and use the event list with the list of UI elements associated with each event to dynamically exercise the events at runtime in order to explore the applications' behaviour.

1.3 Research Objectives

The aim of this study is to come up with a hybrid approach for reverse engineering mobile applications for model generation for automated testing. To achieve this aim, four objectives were identified as follows:

i. To propose an improved technique for the systematic exploration of Android mobile apps.

ii. To develop an improved GUI crawling technique for mobile apps based on the approach in (i).

iii. To implement the technique in Automated Model Generator for Android (AMOGA) tool for reverse engineering GUI model from mobile applications.

iv. To validate the proposed technique by measuring the coverage achieved and time for the exploration, and compare the results with the results of existing techniques for automated model generation.

1.4 Research Scope

The research scope is drawn based on the research objectives presented above. As every research is delimited by time and boundary in order to be feasible within the time frame, the scope of the research is stated below:

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

Mobile applications are usually developed in different platforms (Android, iOS, Windows mobile and Blackberry) using different programming languages. They are broadly categorized into three types as shown in Figure 1.1. The native apps are developed specifically for a particular platform and they run on the device's operating system. Web-based applications run on the web browser on a mobile device while the hybrid applications are partly native and partly web apps, and allow cross-platform development. The native applications are more popular among developers because they have the capability to utilize full advantages of the native features on a device.

Owing to the fact that android device has become more popular among other platforms due its open source nature, this research focuses on native and hybrid applications designed for the Android device. The research is focused on model generation from mobile apps which can be used for model-based testing. The model of interest is Finite State Machine (FSM) model that represent the behaviour of an app. An evaluation of the technique is performed to ascertain the quality of the generated model.

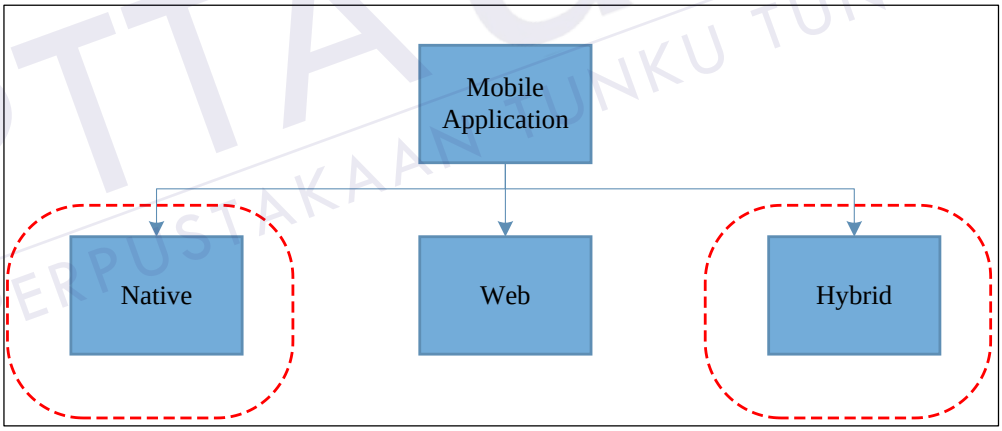

Figure 1.1: Mobile applications considered

The data to be used for the evaluation of the proposed technique is Android mobile apps and the source of the applications is Google play store. The proposed approach is also evaluated based on code coverage and execution time, and it is compared with the existing techniques for automated model generation.

1.5 Significance of Research

The application stores have multiple choices of apps to be downloaded by the users. This has made users intolerable with unstable application and un-install it. In view of this, the quality and performance of an application is significant. The benefits that can be derived from this research is the development of a technique and a tool to help developers and testers in testing mobile apps in order to improve the quality and performance of their apps. This is through automated generation of models from mobile apps which can be used for constructing test cases to support testing of mobile apps. This is because testing apps with significant coverage can help in improving the quality and performance of applications.

1.6 Expected Outcome

Reverse engineering involves two main processes. First, is the extraction of information from an application and secondly, representing the information in a format the users can easily understand. Several abstractions can be derived from the extracted information depending on the purpose. At the end of this research, a tool will be developed based on the proposed approach that will aid the automated generation of high quality GUI models from an application that describes the behaviour of an application. The model describes the state of events which can be used for the generation of test scripts for testing an app.

1.7 Structure of the Thesis

The thesis is structured as follows. Chapter 1 presents the background of the study, research motivation, challenges of testing mobile apps, and objectives of this research. It further discussed the scope, significance and expected outcome of the research.

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

REFERENCES

[1] A. K. Karlson, B. R. Meyers, A. Jacobs, P. Johns, and S. K. Kane, "Working overtime: Patterns of smartphone and PC usage in the day of an information worker," in *Pervasive computing*, ed: Springer, 2009, pp. 398-405.

[2] L. Fortunati and S. Taipale, "The advanced use of mobile phones in five European countries," *The British journal of sociology*, vol. 65, pp. 317-337, 2014.

[3] I. Gartner, ""Worldwide Smartphone Sales", February, 2017," *http://www.gartner.com/newsroom/id/3609817*. Accessed February, 2017.

[4] W. Yang, M. R. Prasad, and T. Xie, "A grey-box approach for automated GUI-model generation of mobile applications," in *Fundamental Approaches to Software Engineering*, ed: Springer, 2013, pp. 250-265.

[5] R. Murtagh, "The Biggest Shift Since the Internet Began," *https://searchenginewatch.com/sew/opinion/2353616/mobile-now-exceeds-pc-the-biggest-shift-since-the-internet-began*. Accessed February, 2017.

[6] F. Nayebi, J.-M. Desharnais, and A. Abran, "The state of the art of mobile application usability evaluation," in *CCECE*, 2012, pp. 1-4.

[7] R. Islam, R. Islam, and T. Mazumder, "Mobile application and its global impact," *International Journal of Engineering & Technology (IJEST)*, 2010.

[8] R. Minelli and M. Lanza, "Software Analytics for Mobile Applications-Insights & Lessons Learned," in *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on Software Maintenance and Reengineering*, 2013, pp. 144-153.

[9] I. Gartner, "Mobile Apps Will Be a Vehicle for Cognizant Computing. August, 2016," *http://www.gartner.com/newsroom/id/2654115*. Accessed February, 2017.

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

[10] D. Amalfitano, A. R. Fasolino, and P. Tramontana, "A gui crawling-based technique for android mobile application testing," in *Fourth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2011, pp. 252-261.

[11] A. I. Wasserman, "Software engineering issues for mobile application development," in *Proceedings of the FSE/SDP workshop on Future of software engineering research*, 2010, pp. 397-400.

[12] J. Dehlinger and J. Dixon, "Mobile application software engineering: Challenges and research directions," in *Workshop on Mobile Software Engineering*, 2011.

[13] F. Balagtas-Fernandez, J. Forrai, and H. Hussmann, "Evaluation of user interface design and input methods for applications on mobile touch screen devices," *Human-Computer Interaction–INTERACT 2009*, pp. 243-246, 2009.

[14] H. Muccini, A. Di Francesco, and P. Esposito, "Software testing of mobile applications: Challenges and future research directions," in *7th International Workshop on Automation of Software Test (AST)*,, 2012, pp. 29-35.

[15] É. Payet and F. Spoto, "Static analysis of Android programs," *Information and Software Technology*, vol. 54, pp. 1192-1201, 2012.

[16] C. Hu and I. Neamtiu, "Automating GUI testing for Android applications," in *Proceedings of the 6th International Workshop on Automation of Software Test*, 2011, pp. 77-83.

[17] P. Bhattacharya, L. Ulanova, I. Neamtiu, and S. C. Koduru, "An empirical analysis of bug reports and bug fixing in open source android apps," in *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*, 2013, pp. 133-143.

[18] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, *et al.*, "TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, p. 5, 2014.

[19] A. Rountev and D. Yan, "Static Reference Analysis for GUI Objects in Android Software," presented at the Proceedings of Annual IEEE/ACM International Symposium on Code Generation and Optimization, Orlando, FL, USA, 2014.

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

[20] T. Azim and I. Neamtiu, "Targeted and depth-first exploration for systematic testing of android apps," in *Proceedings of the 2013 ACM SIGPLAN international conference on Object oriented programming systems languages & applications*, 2013, pp. 641-660.

[21] S. Yang, D. Yan, H. Wu, Y. Wang, and A. Rountev, "Static control-flow analysis of user-driven callbacks in Android applications," in *International Conference on Software Engineering (ICSE)*, 2015.

[22] L. Osterweil, "Strategic directions in software quality," *ACM Computing Surveys (CSUR)*, vol. 28, pp. 738-750, 1996.

[23] A. Méndez-Porras, C. Quesada-López, and M. Jenkins, "Automated testing of mobile applications: a systematic map and review," in *XVIII Ibero-American Conference on Software Engineering, Lima-Peru*, 2015, pp. 195-208.

[24] M. Janicki, M. Katara, and T. Pääkkönen, "Obstacles and opportunities in deploying model- based GUI testing of mobile software: a survey," *Software Testing, Verification and Reliability*, vol. 22, pp. 313-341, 2012.

[25] S. Salva, P. Laurençot, and S. R. Zafimiharisoa, "Model Inference of Mobile Applications with Dynamic State Abstraction," in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, ed: Springer, 2016, pp. 177-193.

[26] M. Young, *Software testing and analysis: process, principles, and techniques*: John Wiley & Sons, 2008.

[27] A. Kull, "Automatic GUI Model Generation: State of the Art," in *Software Reliability Engineering Workshops (ISSREW), 2012 IEEE 23rd International Symposium on*, 2012, pp. 207-212.

[28] L. Lu and Y. Huang, "Automated GUI Test Case Generation," in *Computer Science & Service System (CSSS), 2012 International Conference on*, 2012, pp. 582-585.

[29] T. Cipresso and M. Stamp, "Software Reverse Engineering," in *Handbook of Information and Communication Security*, ed: Springer, 2010, pp. 659-696.

[30] J. Krijnen, "Software Reverse Engineering," 2013.

[31] G. Canfora, M. D. Penta, and L. Cerulo, "Achievements and challenges in software reverse engineering," *Commun. ACM*, vol. 54, pp. 142-151, 2011.

[32] P. Kabbash, W. Buxton, and A. Sellen, "Two-handed input in a compound task," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1994, pp. 417-423.

[33] P. Gerrard, "Testing GUI applications," *EuroSTAR*, vol. 97, pp. 24-28, 1997.

[34] Q. Xie, "Developing cost-effective model-based techniques for GUI testing," in *Proceedings of the 28th international conference on Software engineering*, 2006, pp. 997-1000.

[35] I. C. Morgado, A. C. Paiva, and J. P. Faria, "Automated pattern-based testing of mobile applications," in *Quality of Information and Communications Technology (QUATIC), 2014 9th International Conference on the*, 2014, pp. 294-299.

[36] D. Amalfitano, A. R. Fasolino, P. Tramontana, S. D. Carmine, and A. M. Memon, "Using GUI ripping for automated testing of Android applications," presented at the Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, Essen, Germany, 2012.

[37] A. M. Memon, "An event-flow model of GUI-based applications for testing," *Software Testing Verification and Reliability*, vol. 17, pp. 137-158, 2007.

[38] A. P. Grilo, A. R. Paiva, and J. P. Faria, "Reverse engineering of GUI models for testing," in *Iberian Conference on Information Systems and Technologies (CISTI), 2010 5th* 2010, pp. 1-6.

[39] "Android GUITAR," *https://sourceforge.net/projects/guitar/*. Accessed November, 2016.

[40] D. Amalfitano, A. R. Fasolino, P. Tramontana, S. De Carmine, and G. Imparato, "A toolset for GUI testing of Android applications," in *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, 2012, pp. 650-653.

[41] S. Salva and S. R. Zafimiharisoa, "Model Reverse-engineering of Mobile Applications with Exploration Strategies," *In Proceedings of the 9th International Conference on Software Engineering Advances (ICSEA), October 12-16, 2014, Nice, France.*, 2014.

[42] I. Coimbra Morgado, A. C. Paiva, and J. Pascoal Faria, "Dynamic Reverse Engineering of Graphical User Interfaces," *International Journal On Advances in Software*, vol. 5, pp. 224-236, 2012.

[43] C. E. Silva and J. C. Campos, "Combining static and dynamic analysis for the reverse engineering of web applications," presented at the Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems, London, United Kingdom, 2013.

[44] T. Systä, "Static and dynamic reverse engineering techniques for Java software systems," 2000.

[45] S. Yang, H. Zhang, H. Wu, Y. Wang, D. Yan, and A. Rountev, "Static Window Transition Graphs for Android (T)," presented at the Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2015.

[46] S. Yang, "Static analyses of GUI behavior in Android applications," The Ohio State University, 2015.

[47] B. Nguyen, B. Robbins, I. Banerjee, and A. Memon, "GUITAR: an innovative tool for automated testing of GUI-driven software," *Automated Software Engineering*, vol. 21, pp. 65-105, 2014/03/01 2014.

[48] S. R. Choudhary, A. Gorla, and A. Orso, "Automated Test Input Generation for Android: Are We There Yet?(E)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2015, pp. 429-440.

[49] "Android Architecture," *https://www.tutorialspoint.com/android/android_architecture.htm*. Accessed February, 2017.

[50] T.-H. Su, H.-J. Tsai, K.-H. Yang, P.-C. Chang, T.-F. Chen, and Y.-T. Zhao, "Reconfigurable vertical profiling framework for the android runtime system," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 13, p. 59, 2014.

[51] A. Memon, I. Banerjee, and A. Nagarajan, "GUI ripping: Reverse engineering of graphical user interfaces for testing," in *10th Working Conference on Reverse Engineering (WCRE 2003)*, 2003, pp. 260-260.

[52] E. Masi, G. Cantone, M. Mastrofini, G. Calavaro, and P. Subiaco, "Mobile apps development: A framework for technology decision making," in *Mobile Computing, Applications, and Services*, ed: Springer, 2013, pp. 64-79.

[53] M. E. Joorabchi, A. Mesbah, and P. Kruchten, "Real challenges in mobile app development," in *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on*, 2013, pp. 15-24.

[54] "Android Developers Site," *https://developer.android.com/guide/components/*

index.html. Accessed February, 2017.

[55] A. G. Parada and L. B. de Brisolara, "A Model Driven Approach for Android Applications Development," in *Computing System Engineering (SBESC), 2012 Brazilian Symposium on*, 2012, pp. 192-197.

[56] "Google, Activity," *https://developer.android.com/reference/android/app/Activity.html*. Accessed February, 2017.

[57] P. C. Jorgensen, *Software testing: a craftsman's approach*: CRC press, 2013.

[58] M. J. Harrold, "Testing: a roadmap," in *Proceedings of the Conference on the Future of Software Engineering*, 2000, pp. 61-72.

[59] I. Banerjee, B. Nguyen, V. Garousi, and A. Memon, "Graphical user interface (GUI) testing: Systematic mapping and repository," *Information and Software Technology*, vol. 55, pp. 1679-1694, 2013.

[60] A. M. Memon, M. E. Pollack, and M. L. Soffa, "Hierarchical GUI test case generation using automated planning," *IEEE transactions on software engineering*, vol. 27, pp. 144-155, 2001.

[61] X. Yuan and A. M. Memon, "Generating event sequence-based test cases using GUI runtime state feedback," *IEEE Transactions on Software Engineering*, vol. 36, pp. 81-95, 2010.

[62] X. Yuan and A. M. Memon, "Using GUI run-time state as feedback to generate test cases," in *29th International Conference on Software Engineering (ICSE'07)*, 2007, pp. 396-405.

[63] D. Huizinga and A. Kolawa, *Automated defect prevention: best practices in software management*: John Wiley & Sons, 2007.

[64] P. Aho, M. Suarez, A. Memon, and T. Kanstrén, "Making GUI Testing Practical: Bridging the Gaps," in *Information Technology - New Generations (ITNG), 2015 12th International Conference on*, 2015, pp. 439-444.

[65] R. Kumar and R. G. Bhatia, "Interaction Based Software Testing," 2010.

[66] G. de Cleva Farto and A. T. Endo, "Evaluating the model-based testing approach in the context of mobile applications," *Electronic notes in Theoretical computer science*, vol. 314, pp. 3-21, 2015.

[67] M. Utting and B. Legeard, *Practical model-based testing: a tools approach*: Morgan Kaufmann, 2010.

[68] I. K. El- Far and J. A. Whittaker, "Model- Based Software Testing," *Encyclopedia of Software Engineering*, 2001.

[69] I. C. Morgado, A. Paiva, and J. P. Faria, "Reverse engineering of graphical user interfaces," in *The Sixth International Conference on Software Engineering Advances, Barcelona*, 2011, pp. 293-298.

[70] P. Aho, T. Raty, and N. Menz, "Dynamic reverse engineering of GUI models for testing," in *Control, Decision and Information Technologies (CoDIT), 2013 International Conference on*, 2013, pp. 441-447.

[71] M. E. Joorabchi and A. Mesbah, "Reverse engineering iOS mobile applications," in *Reverse Engineering (WCRE), 2012 19th Working Conference on*, 2012, pp. 177-186.

[72] I. Coimbra Morgado, A. Paiva, and J. Pascoal Faria, "Reverse engineering of graphical user interfaces," in *ICSEA 2011, The Sixth International Conference on Software Engineering Advances*, 2011, pp. 293-298.

[73] J. C. Campos, J. Saraiva, C. Silva, and J. C. Silva, "GUIsurfer: A Reverse Engineering Framework for User Interface Software," *Reverse Engineering-Recent Advances and Applications*, pp. 31-54, 2012.

[74] M. M. Moore, "Rule-based detection for reverse engineering user interfaces," in *Reverse Engineering, 1996., Proceedings of the Third Working Conference on*, 1996, pp. 42-48.

[75] J. Křoustek and D. Kolář, "Approaching Retargetable Static, Dynamic, and Hybrid Executable-Code Analysis," *Acta Informatica Pragensia*, vol. 2, pp. 18-29, 2013.

[76] J. K. Y. Ng, Y. G. Guéhéneuc, and G. Antoniol, "Identification of behavioural and creational design motifs through dynamic analysis," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 22, pp. 597-627, 2010.

[77] P. Aho, M. Suarez, T. Kanstren, and A. M. Memon, "Murphy Tools: Utilizing Extracted GUI Models for Industrial Software Testing," in *Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on*, 2014, pp. 343-348.

[78] S. Yang, D. Yan, and A. Rountev, "Testing for poor responsiveness in Android applications," in *Engineering of Mobile-Enabled Systems (MOBS), 2013 1st International Workshop on the*, 2013, pp. 1-6.

[79] J. C. Silva, C. Silva, R. D. Gonçalo, J. Saraiva, and J. C. Campos, "The GUISurfer tool: towards a language independent approach to reverse

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

engineering GUI code," in *Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems*, Berlin, Germany, 2010, pp. 181-186.

[80] S. Staiger, "Reverse Engineering of Graphical User Interfaces Using Static Analyses," presented at the Proceedings of the 14th Working Conference on Reverse Engineering, 2007.

[81] S. Arlt, A. Podelski, C. Bertolini, M. Schaf, I. Banerjee, and A. M. Memon, "Lightweight static analysis for GUI testing," in *Software Reliability Engineering (ISSRE), 2012 IEEE 23rd International Symposium on*, 2012, pp. 301-310.

[82] A. R. Paiva, J. P. Faria, and P. C. Mendes, "Reverse Engineered Formal Models for GUI Testing," in *Formal Methods for Industrial Critical Systems*. vol. 4916, S. Leue and P. Merino, Eds., ed: Springer Berlin Heidelberg, 2008, pp. 218-233.

[83] A. Mesbah, A. van Deursen, and S. Lenselink, "Crawling Ajax-Based Web Applications through Dynamic Analysis of User Interface State Changes," *ACM Trans. Web*, vol. 6, pp. 1-30, 2012.

[84] S. Demeyer, S. Ducasse, and M. Lanza, "A hybrid reverse engineering approach combining metrics and program visualisation," in *Reverse Engineering, 1999. Proceedings. Sixth Working Conference on*, 1999, pp. 175-186.

[85] A. Memon, "AndroidGUITAR," *http://sourceforge.net/apps/mediawiki/guitar/index.php?title=Android_GUITAR*. Accessed November, 2016.

[86] M. Yuan and Y. Xuebing, "An FSM based GUI test automation model," in *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*, 2010, pp. 120-126.

[87] W. Choi, G. Necula, and K. Sen, "Guided gui testing of android apps with minimal restart and approximate learning," in *ACM SIGPLAN Notices*, 2013, pp. 623-640.

[88] E. Shah and E. Tilevich, "Reverse-engineering user interfaces to facilitateporting to and across mobile devices and platforms," in *Proceedings of the compilation of the co-located workshops on DSM'11, TMC'11, AGERE!'11, AOOPES'11, NEAT'11, & VMIL'11*, 2011, pp. 255-260.

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH

[89] T. Takala, M. Katara, and J. Harty, "Experiences of system-level model-based GUI testing of an Android application," in *Software Testing, Verification and Validation (ICST), 2011 IEEE Fourth International Conference on*, 2011, pp. 377-386.

[90] D. Amalfitano, A. R. Fasolino, and P. Tramontana, "An Iterative Approach for the Reverse Engineering of Rich Internet Application User Interfaces," in *Fifth International Conference on Internet and Web Applications and Services (ICIW), 2010* 2010, pp. 401-410.

[91] B. Johnson, Y. Song, E. Murphy-Hill, and R. Bowdidge, "Why don't software developers use static analysis tools to find bugs?," in *Software Engineering (ICSE), 2013 35th International Conference on*, 2013, pp. 672-681.

[92] B. Wichmann, A. Canning, D. Clutterbuck, L. Winsborrow, N. Ward, and D. Marsh, "Industrial perspective on static analysis," *Software Engineering Journal*, vol. 10, pp. 69-75, 1995.

[93] G. Horváth and N. Pataki, "Source language representation of function summaries in static analysis," in *Proceedings of the 11th Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems*, 2016, p. 6.

[94] F. E. Allen, "Control flow analysis," in *ACM Sigplan Notices*, 1970, pp. 1-19.

[95] A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers, Principles, Techniques*: Addison wesley, 1986.

[96] "IBM WALA framework," *https://github.com/jgf/joana*. Accessed February, 2017.

[97] A. P. Fuchs, A. Chaudhuri, and J. S. Foster, "Scandroid: Automated security certification of android applications," *Manuscript, Univ. of Maryland, http://www. cs. umd. edu/avik/projects/scandroidascaa*, vol. 2, 2009.

[98] J. Jeon and J. S. Foster, "Troyd: Integration Testing for Android," 2012.

[99] J. Wu, P. Teregowda, J. P. Fern, n. Ram, *et al.*, "The evolution of a crawling strategy for an academic document search engine: whitelists and blacklists," presented at the Proceedings of the 4th Annual ACM Web Science Conference, Evanston, Illinois, 2012.

[100] D. Shestakov, "Current challenges in web crawling," in *International Conference on Web Engineering*, 2013, pp. 518-521.

[101] J. Kleinberg and É. Tardos, *Algorithm design*: Pearson Education India, 2006.

[102] R. Feldt and A. Magazinius, "Validity Threats in Empirical Software Engineering Research-An Initial Survey," in *SEKE*, 2010, pp. 374-379.

[103] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*: Springer Science & Business Media, 2012.

[104] J. A. Maxwell, *Qualitative research design: An interactive approach* vol. 41: Sage publications, 2012.

[105] T. D. Cook, D. T. Campbell, and A. Day, *Quasi-experimentation: Design & analysis issues for field settings* vol. 351: Houghton Mifflin Boston, 1979.

[106] S. Leshem and V. Trafford, "Overlooking the conceptual framework," *Innovations in Education and Teaching International,* vol. 44, pp. 93-105, 2007/02/01 2007.

[107] S. Yang, H. Zhang, H. Wu, Y. Wang, D. Yan, and A. Rountev, "GATOR: Program Analysis Toolkit For Android," *http://web.cse.ohio-state.edu/presto/software/gator/*. Accessed February, 2017.

[108] E. Merlo, P.-Y. Gagné, J.-F. Girard, K. Kontogiannis, L. Hendren, P. Panangaden, *et al.*, "Reengineering user interfaces," *IEEE Software,* vol. 12, pp. 64-73, 1995.

[109] A. M. Memon, "A comprehensive framework for testing graphical user interfaces," University of Pittsburgh, 2001.

[110] H. Zhu and P. A. Hall, "Test data adequacy measurement," *Software Engineering Journal,* vol. 8, pp. 21-29, 1993.

[111] Emma, "An open source Java code coverage tool," *http://emma.sourceforge.net/*. Accessed February, 2017.

[112] A. Bartel, J. Klein, Y. Le Traon, and M. Monperrus, "Dexpler: converting android dalvik bytecode to jimple for static analysis with soot," in *Proceedings of the ACM SIGPLAN International Workshop on State of the Art in Java Program analysis*, 2012, pp. 27-38.

[113] B. Alll and C. Tumbleson, "Dex2jar: Tools to work with android.dex and java.class files," ed.

[114] U. Apache, "Robotium," *http://code.google.com/p/robotium*. Accessed February, 2017.

[115] R. d. F. da Computaçao, "GUISURFER: A Generic Framework for Reverse Engineering of Graphical User Interfaces," Universidade do Minho, 2010.

[116] R. J. Jacob, "Using formal specifications in the design of a human-computer interface," *Communications of the ACM,* vol. 26, pp. 259-264, 1983.

[117] "Soot Program Analysis Framework," *http://www.sable.mcgill.ca/soot/*. Accessed February, 2017.

[118] T. J. Misa and P. L. Frana, "An interview with Edsger W. Dijkstra," *Commun. ACM,* vol. 53, pp. 41-47, 2010.

[119] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik,* vol. 1, pp. 269-271, December 01 1959.

[120] K. Mehlhorn and P. Sanders, *Algorithms and data structures: The basic toolbox*: Springer Science & Business Media, 2008.

[121] H. P. Breivold, I. Crnkovic, and M. Larsson, "A systematic review of software architecture evolution research," *Information and Software Technology,* vol. 54, pp. 16-40, 2012.

[122] H. Reza, S. Endapally, and E. Grant, "A model-based approach for testing gui using hierarchical predicate transition nets," in *Information Technology, 2007. ITNG'07. Fourth International Conference on*, 2007, pp. 366-370.

[123] GoogleCode, "Monkey: UI Application Exerciser," *http://developer.android.com/guide/developing/tools/monkey.html*. Accessed February, 2017.

[124] D. Amalfitano, A. R. Fasolino, P. Tramontana, B. D. Ta, and A. M. Memon, "MobiGUITAR: Automated model-based testing of mobile apps," *IEEE Software,* vol. 32, pp. 53-59, 2015.

PTTA UTHM PERPUSTAKAAN TUNKU TUN AMINAH